

Ajax

Mario Schwede

Großes Seminar
Fachhochschule Gießen-Friedberg

20.02.2007

Inhaltsverzeichnis:

- 1 Einführung
- 2 Benutzung
- 3 Anfrage an den Server
- 4 Antwort vom Server empfangen
- 5 Frameworks
- 6 Diskussion
- 7 eStudy
- 8 Literatur
- 9 Weblinks

Allgemein:

- **Asynchronous JavaScript and XML.**
- Desktop-ähnliches Verhalten von Webanwendungen.
- Keine neue Technik.
- Unterstützung für GUI-Elemente.
- Kombination aus:
 - HTML
 - Javascript
 - Text, XML oder JSON
 - Einer serverseitigen Websprache
 - DOM

Besonderheiten:

- Kein Warten auf die Antwort vom Server.
- Kein Seitenreload.
- Ständiger Datenaustausch möglich.
- Nur Nutzdatenaustausch.

Model-View-Controller als Verständnis-Hilfe:

- Trennung der Bestandteile in:
 - View
 - Controller
 - Model

View:

- Besteht aus HTML.
- Ruft Funktionen des Controllers auf.
- Stellt die Antwort dar.

Controller:

- Besteht aus Javascript.
- Schickt Anfragen an das Model.
- Sendet und empfängt über ein XMLHttpRequest-Objekt.
- Bekommt die Antwort vom Model.
- Verändert die View mit Hilfe von DOM.

Model:

- Besteht aus einer beliebigen serverseitigen Websprache.
- Bearbeitet die Anfrage vom Controller.
- Schickt eine Antwort zurück.
- Kodiert die Antwort im Text-, XML- oder JSON-Format.

Vorgang:

- 1 Erstelle eine globale Instanz vom XMLHttpRequest-Objekt.
- 2 Versehe die zu übertragene Werte im HTML-Code mit ID-Tags.
- 3 Erstelle eine POST oder GET Anfrage-Funktion in Javascript.
- 4 Verknüpfe ein Event mit der Anfrage-Funktion.

Globale Instanz erstellen:

```
var request = null;
try {
    request = new XMLHttpRequest();
}
catch (trymicrosoft) {
    try {
        request = new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (othermicrosoft) {
        try {
            request = new ActiveXObject("Microsoft.XMLHTTP");
        }
        catch (failed) {
            request = null;
        }
    }
}
if (request == null)
    alert("Error creating request object!");
```

HTML-Seite erstellen:

```
<html>
<head>
  <title>Erste Schritte mit Ajax</title>
  <script type="text/javascript" src="ajax.js"> </script>
  <script type="text/javascript" src="spezificCode.js"> </script>
</head>

<body>
  Wert:<span id="myValue">2000</span>

  <form method="POST"> <!-- <form method="GET"> -->
    <input value="Mach was" type="button" onClick="submitSomething();" />
  </form>
</body>
</html>
```

POST Anfrage-Funktion erstellen:

```
function submitSomething()  
{  
    var toPost = Document.getElementById("myValue").value;  
  
    var url = "myServerScript.php";  
  
    request.open("POST", url, true);  
  
    request.onreadystatechange = callBackFunction;  
  
    request.setRequestHeader  
    ("Content-Type", "application/x-www-form-urlencoded");  
  
    request.send("myValue=" + escape(toPost));  
}
```

GET Anfrage-Funktion erstellen:

```
function submitSomething()  
{  
    var toGet = document.getElementById("myValue").value;  
    var url = "myServerScript.php?myValue=" + escape(toGet);  
    request.open("GET", url, true);  
    request.onreadystatechange = callBackFunction;  
    request.setRequestHeader  
    ("Content-Type", "application/x-www-form-urlencoded");  
    request.send (null);  
}
```

Antwort empfangen:

Vorgang:

- 1 Erstelle Serverscript.
- 2 Erstelle in Javascript eine Callback-Funktion

Datenformate:

- Text
- XML
- JSON

Serverscript mit Text-Antwort:

```
<?php
  // Lese Parameter ein
  // Mache etwas!
  // Speichere das Ergebnis in erg
  // Gebe erg zurück

  echo $erg;
?>
```

Callback-Funktion für Text-Format:

```
function callBackFunction ()
{
    if (request.readyState == 4)
    {
        if (request.status == 200)
        {
            var response = request.responseText;

            // mache was!
            // Darstellung des Ergebnisses mit Hilfe von DOM.
        }
        else
        {
            var message = request.getResponseHeader("Status");
            if ((message == null) || (message.length <= 0))
            {
                alert("Error! Request status is " + request.status);
            }
        }
    }
}
```


Eigenschaften vom Text-Format:

- Leichtgewichtiges Format.
- Einfache Verarbeitung vom Client.
- Einfache Bereitstellung vom Server.
- Nur ein Wert übertragbar.

Serverscript mit XML-Antwort:

```
<?php
// Lese Parameter ein
// Mache etwas!
// Speichere die Ergebnisse in den Variablen
// nachname, vorname, strasse
// Gebe die Adresse gebündelt als XML zurück

header("Content-Type: text/xml");
echo "<?xml version='1.0' encoding='utf-8'?>";
?>

<adresse>
  <nachname><? echo $nachname; ?></ nachname >
  <vorname><? echo $vorname; ?></ vorname >
  <strasse><? echo $strasse; ?></strasse>
</adresse>
```

Callback-Funktion für XML-Format:

```
function callBackFunction () {  
    if (request.readyState == 4)  
    {  
        if (request.status == 200) {  
            var xmlResponse = request.responseXML;  
  
            var xmlNachname = xmlResponse.getElementsByTagName("nachname")[0];  
            var nachname = xmlNachname.firstChild.nodeValue;  
  
            var xmlVorname = xmlResponse.getElementsByTagName("vorname")[0];  
            var vorname = xmlVorname.firstChild.nodeValue;  
  
            var xmlStrasse = xmlResponse.getElementsByTagName("strasse")[0];  
            var strasse = xmlStrasse.firstChild.nodeValue;  
  
            // Mit DOM die Oberfläche verändern  
        } else {  
            // Fehlerbehandlung  
        }  
    }  
}
```

Eigenschaften vom XML-Format:

- Hoher Overhead.
- Umständliche Verarbeitung vom Client.
- Einfache Bereitstellung vom Server.
- Beliebig viele Werte übertragbar.
- Standardisiert.

Serverscript mit JSON-Antwort und Bibliothek:

```
<?php
require_once( 'JSON.php' );

// mache was!

$json = new Services_JSON();

$adresse = array( 'nachname' => $nachname ,
                  'vorname'  => $vorname ,
                  'strasse'  => $strasse );

$output = $json->encode( $adresse );
print( $output );
?>
```

Serverscript mit JSON-Antwort ohne Bibliothek:

```
<?php  
  
// mach was!  
  
$adresse = '{ "adresse": [  
            {  
                "nachname": ' + $nachname +  
                "vorname": ' + $vorname +  
                "strasse": ' + $strasse +  
            }  
        ] }';  
print( $adresse );  
  
?>
```

Callback-Funktion für JSON-Format:

```
function callBackFunction () {  
    if (request.readyState == 4)  
    {  
        if (request.status == 200)  
        {  
            // JSON Antwort empfangen  
  
            var jsonResponse = eval('(' + request.responseText + ')');  
  
            var nachname = jsonResponse.adresse[0].nachname;  
            var vorname = jsonResponse.adresse[0].vorname;  
            var strasse = jsonResponse.adresse[0].strasse;  
  
            // Mit DOM Oberfläche verändern  
        }  
        else  
        {  
            // Fehlerbehandlung  
        }  
    }  
}
```

Eigenschaften vom JSON-Format:

- Geringer Overhead.
- Einfache Verarbeitung vom Client.
- Komplizierte Bereitstellung vom Server.
- Beliebig viele Werte übertragbar.

Frameworks:

Aufgabe:

- Vereinfachung der Anwendungsentwicklung.
- Browser-Eigenarten vor dem Programmierer verstecken.
- Grafische Elemente bereitstellen.

Unterteilung:

- Clientseitige Frameworks
- Serverseitige Frameworks

Clientseitige-Frameworks:

Aufgabe:

- Vereinfachung der Erstellung von Javascriptcode zur Nutzung von Ajax.

Frameworks:

- Prototype
- Dojo
- Script.aculo.us
- Rico
- Viele weitere..

Serverseitige-Frameworks:

Aufgabe:

- Möglichst nahtlose Anpassung von Ajax an die Umgebung.
- Weitgehende Befreiung des Programmierens von Javascript.

Es gibt diese Frameworks für viele Websprachen.

Frameworks:

- Sajax
- Xajax
- ASP.NET Ajax 1.0
- Viele weitere..

Diskussion:

Problem:

Wann soll ich Ajax anwenden?

Lösung:

- 1 Gegenüberstellung der Vor- und Nachteile.
- 2 Gewichtung dieser anhand des zu lösenden Problems.

Vorteile:

- Fast nur Nutzdaten Übertragung.
- Kein Seitenreload.
- Ansprechbar der Seite während der Anfrageverarbeitung.
- Ständiger Datenaustausch möglich.
- Keine Browser-Plugin benötigt.
- Viele Frameworks.

Nachteile:

- Lesezeichen sind kaum möglich.
- Zurück-Button kaum möglich.
- Keine Rückmeldung, während der Bearbeitung.
- Gute DOM-Kenntnisse für Entwickler nötig.
- Javascript muss aktiviert sein.
- Die Barrierefreiheit kann darunter leiden.
- Serverlast steigt.

Anwendungsmöglichkeiten:

Wie?

Serverseitiges Framework wie z.B. Sajax

Wann?

- Sehr lang dauernde Operationen.
- Keine Abhängigkeit vom Ergebnis.
- Nur kleinen Änderungen an der Seite.
- Parallele Änderungen während Eingaben.

Literatur:

- Brett D. McLaughlin: *Ajax von Kopf bis Fuß*
Oreilly, 2006, ISBN: 978-3-89721-469-9
- Dave Crane / Eric Pascarella / Darren James: *Ajax in action*
Addison-Wesley , 2006, ISBN: 978-3-8273-2414-6
- Michael Mahemoff: *Ajax Design Patterns*
Oreilly, 2006, ISBN: 978-0-596-10180-0

Weblinks:

- Artikel zu dieser Präsentation:
<https://wiki.estudy-portal.de/index.php/Ajax>
- Artikel in Wikipedia:
[http://de.wikipedia.org/wiki/Ajax\(Programmierung\)](http://de.wikipedia.org/wiki/Ajax(Programmierung))
- Entwicklerhilfe(englisch):
<http://ajaxpatterns.org>
- Linksammlung:
<http://www.drweb.de/weblog/weblog/?p=454>
- Community mit Forum und Blog:
<http://www.ajax-community.de>

Einführung
Benutzung
Anfrage an den Server
Antwort vom Server empfangen
Frameworks
Diskussion
eStudy
Literatur
Weblinks

Danke:

Vielen Dank für die Aufmerksamkeit!